

The Pitfalls Within Duality Views

Philipp Salvisberg
11th March 2026



Timer?

Welcome



Philipp Salvisberg

Founder, Owner and CEO of Grisselbav GmbH

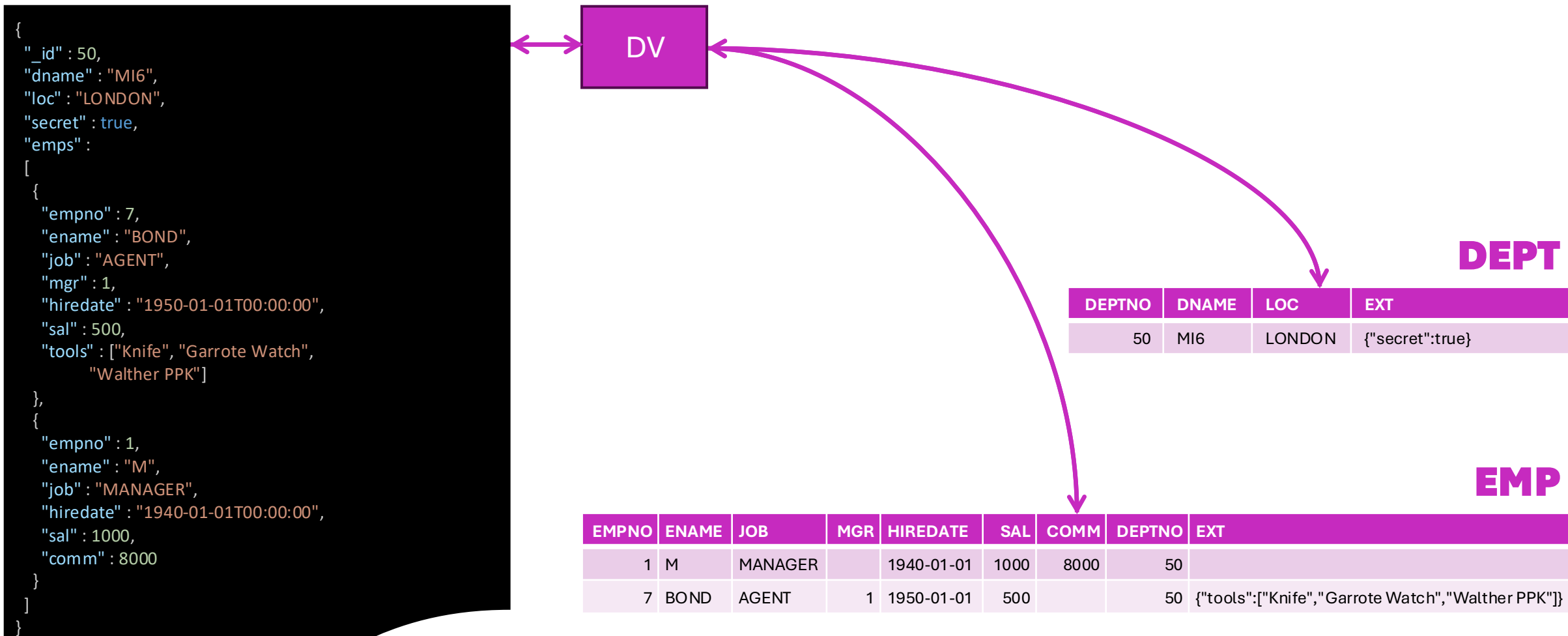
- Database-centric development
- Open-Source development
- Simple & accurate solutions

philipp.salvisberg@grisselbav.com

<https://www.salvis.com/blog/>

What is a DV?

DML via JSON-Relational Duality View (DV)



DV using SELECT

create or replace json duality view dept_dv as

select json {

 '_id': deptno,

 'dname',

 'loc',

 ext as flex,

 'emps':

 (

 select json_arrayagg(

 json {

 emp.empno,

 emp.ename,

 emp.job,

 unnest

 (

 select json {

 'mgr' : mgr.empno with nocheck,

 'mgrname': mgr.ename with nocheck

 }

 from emp mgr

 where mgr.empno = emp.mgr

),

 emp.hiredate,

 emp.sal,

 emp.comm,

 ext as flex

 }

)

 from emp with insert update delete

 where emp.deptno = dept.deptno

)

}

from dept with insert update delete;

Annotations used:
flex, unnest, nocheck,
link (join condition),
insert, update, delete

Applicable
in DVs only

DV using GraphQL

```
create or replace json duality view dept_dv as
dept @insert @update @delete
{
  _id: deptno
  dname
  loc
  ext @flex
  emps: emp @insert @update @delete
  {
    empno
    ename
    job
    emp @unnest @link(from: [mgr])
    {
      mgr : empno @nocheck
      mgrname: ename @nocheck
    }
    hiredate
    sal
    comm
    ext @flex
  }
};
```

All annotations
(directives) start
with '@'

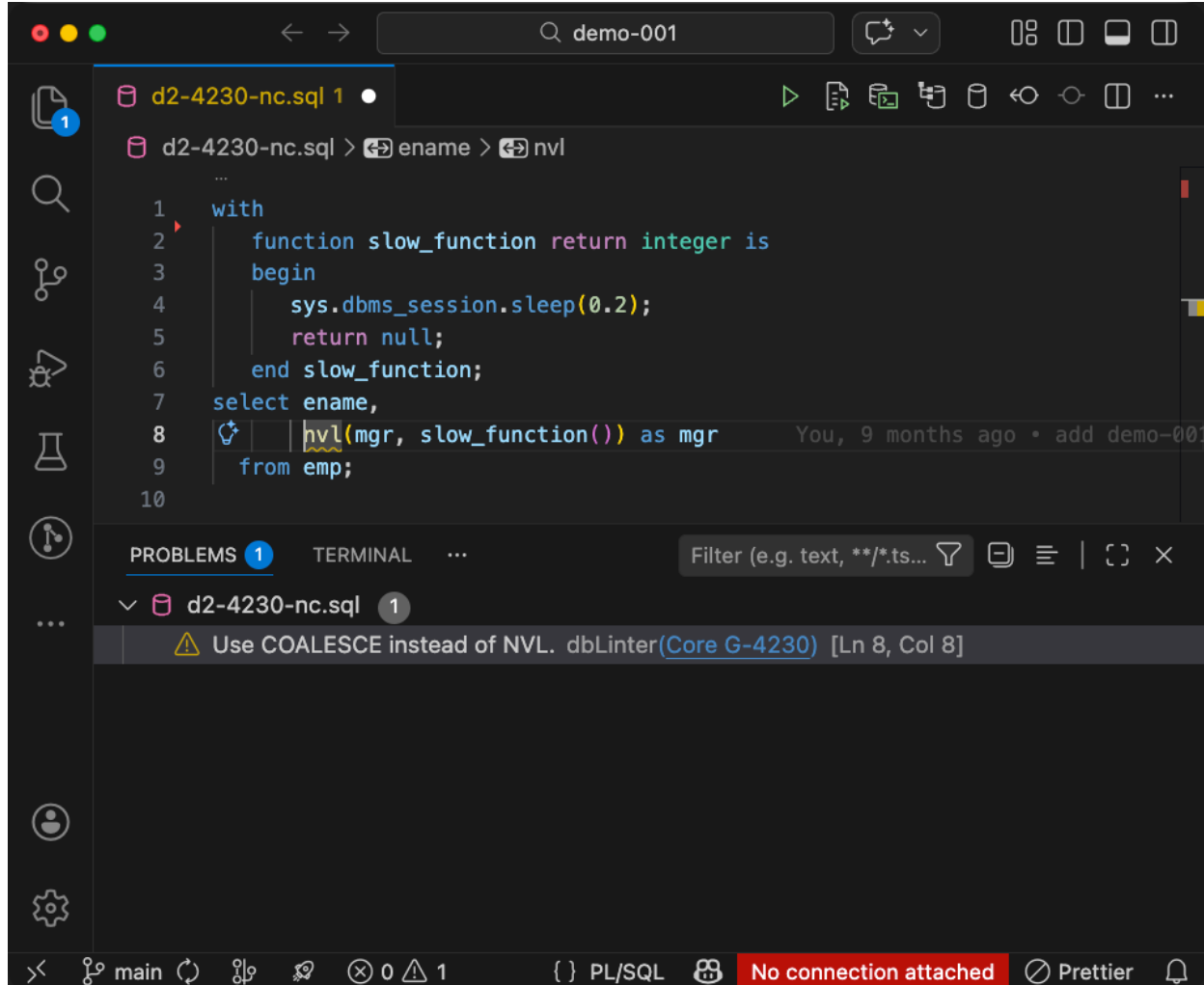
Default Join

Shorter

Looks similar
to the result

Motivation

dbLinter – JSON and DVs



The screenshot shows an IDE window with a dark theme. The main editor displays a SQL script in a file named `d2-4230-nc.sql`. The script contains a PL/SQL block with a function `slow_function` and a query using `nvl`. A linting error is highlighted in the `PROBLEMS` panel at the bottom, indicating that `COALESCE` should be used instead of `NVL`. The status bar at the bottom shows "No connection attached" and "Prettier" is active.

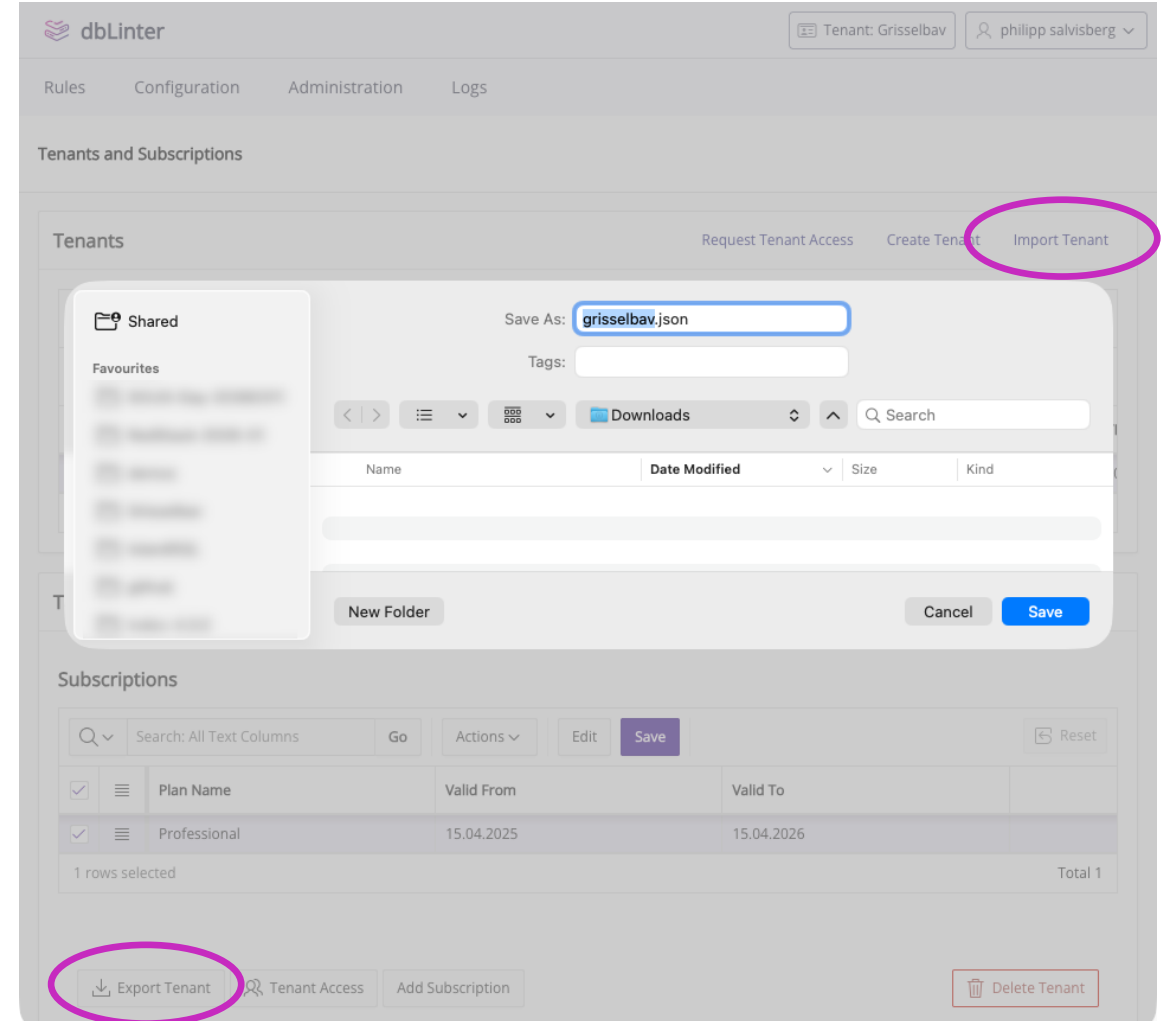
```
1 with
2   function slow_function return integer is
3   begin
4       sys.dbms_session.sleep(0.2);
5       return null;
6   end slow_function;
7 select ename,
8        nvl(mgr, slow_function()) as mgr
9 from emp;
```

PROBLEMS 1 | TERMINAL | ... | Filter (e.g. text, `/**/*.ts...`)

▼ `d2-4230-nc.sql` 1

⚠ Use COALESCE instead of NVL. dbLinter(Core G-4230) [Ln 8, Col 8]

main | 0 | 1 | {} PL/SQL | No connection attached | Prettier



Duality View to Export/Import Tenants

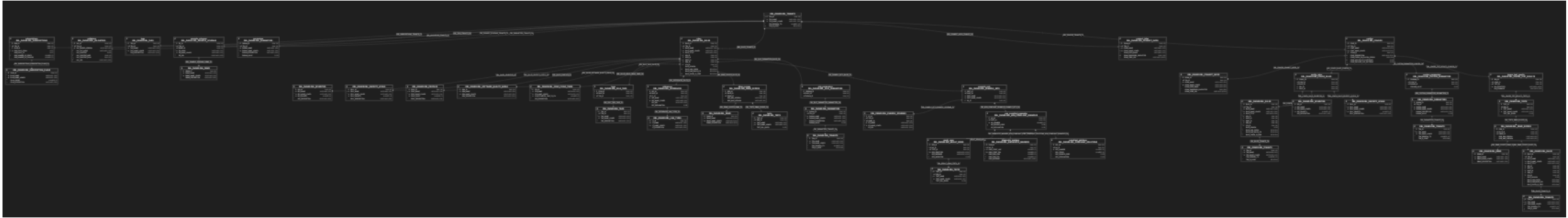
```
create or replace json duality view dbt_tenants_expimp_dv as
dbt_tenants @insert @update @delete
{
  id: ten_id
  ten_name
  ten_session_ttl
  ten_is_core
  subscriptions: dbt_subscriptions @insert @update @delete
  { ~ ins/upd/del requires Product-Admin role
    sub_id
    dbt_subscription_plans @unnest
    {
      plan_id @nocheck
      plan_name @nocheck
    }
    sub_valid_from
    sub_valid_to
    sub_number_of_seats
    sub_number_of_configs
  }
  validators: dbt_validators @insert @update @delete
  {
    val_id
    val_parser_version
    val_file_name
    val_content
    val_content_md5
    val_content_sha1
    val_url
  }
  tags: dbt_tags @insert @update @delete
  {
    tag_id
    tag_name
    tag_description
  }
  example_schemas: dbt_example_schemas
    @insert @update @delete
  {
    es_id
    dbt_dbms @unnest
    {
      dbms_id @nocheck
      dbms_name @nocheck
    }
    es_name
    es_url
  }
  parameters: dbt_parameters @insert @update @delete
  {
    param_id
    param_name
    param_description
    param_value
  }
  rules: dbt_rules @insert @update @delete
  {
    rule_id
    rule_name
    rule_title
    dbt_severities @unnest
    {
      sev_id @nocheck
      sev_name @nocheck
    }
    dbt_severity_levels @unnest
    {
      sev_id @nocheck
      sev_name @nocheck
    }
    dbt_profiles @unnest
    {
      prof_id @nocheck
      prof_name @nocheck
    }
  }
}
```

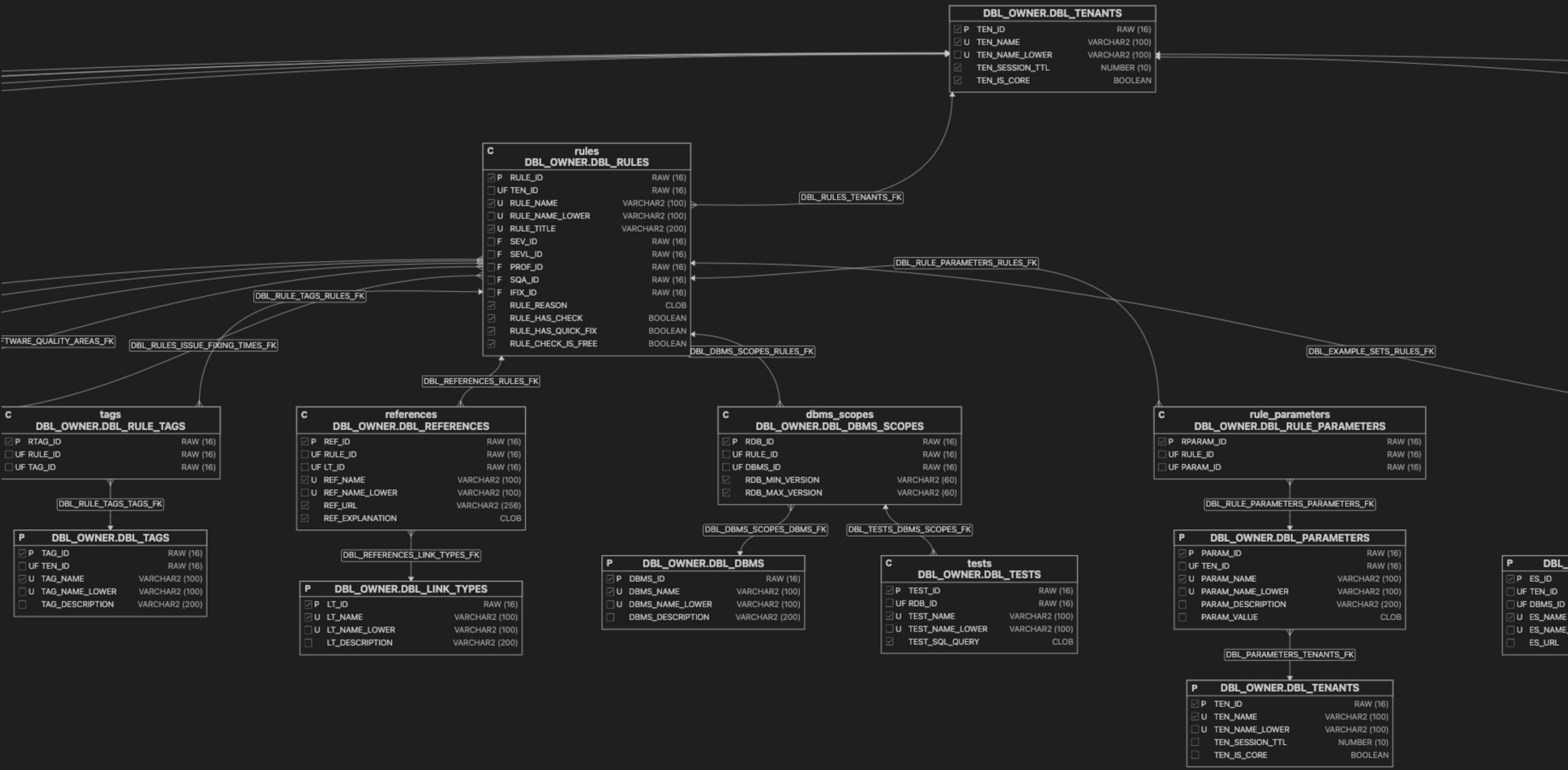
```
dbt_software_quality_areas @unnest
{
  {
    sqs_id @nocheck
    sqs_name @nocheck
  }
  dbt_issue_fixing_times @unnest
  {
    ifix_id @nocheck
    ifix_name @nocheck
  }
  rule_reason
  rule_has_check
  rule_has_quick_fix
  rule_check_is_free
  tags: dbt_rule_tags @insert @update @delete
  {
    rtag_id
    dbt_tags @unnest
    {
      tag_id @nocheck
      tag_name @nocheck
    }
  }
  references: dbt_references
    @insert @update @delete
  {
    ref_id
    dbt_link_types @unnest
    {
      lt_id @nocheck
      lt_name @nocheck
    }
    ref_name
    ref_url
    ref_explanation
  }
  dbms_scopes: dbt_dbms_scopes
    @insert @update @delete
  {
    rdb_id
    dbt_dbms @unnest
    {
      dbms_id @nocheck
      dbms_name @nocheck
    }
    rdb_min_version
    rdb_max_version
    tests: dbt_tests @insert @update @delete
    {
      test_id
      test_name
      test_sql_query
    }
  }
  rule_parameters: dbt_rule_parameters
    @insert @update @delete
  {
    rparam_id
    dbt_parameters @unnest
    {
      param_id @nocheck
      param_name @nocheck
      dbt_tenants @unnest
      {
        ten_id @nocheck
        ten_name @nocheck
      }
    }
  }
  example_sets: dbt_example_sets
    @insert @update @delete
  {
    eset_id
  }
}
```

```
eset_name
dbt_example_schemas @unnest
{
  es_id @nocheck
  es_name @nocheck
}
non_compliant_examples:
dbt_non_compliant_examples @insert @update @delete
{
  nc_id
  nc_number
  nc_source_code
  nc_explanation
  result_rows: dbt_result_rows
    @insert @update @delete
  {
    res_id
    dbt_tests @unnest
    {
      test_id @nocheck
      test_name @nocheck
    }
    res_identifier
    res_message
    res_migration
  }
  diagnostic_markers:
    dbt_diagnostic_markers @insert @update @delete
  {
    diag_id
    diag_start_line
    diag_start_col
    diag_end_line
    diag_end_col
    diag_message
  }
  compliant_solutions:
    dbt_compliant_solutions @insert @update @delete
  {
    sol_id
    sol_number
    sol_stars
    sol_source_code
    sol_explanation
  }
}
connect_infos: dbt_connect_infos @insert @update @delete
{
  conn_id
  conn_name
  conn_user_name
  conn_password_encrypted
  conn_jdbc_url
}
configs: dbt_configs @insert @update @delete
{
  conf_id
  conf_name
  dbt_connect_infos @unnest
  {
    conn_id @nocheck
    conn_name @nocheck
  }
  conf_description
  conf_check_on_syntax_error
  conf_include_file_patterns
  conf_exclude_file_patterns
  config_rules: dbt_config_rules
    @insert @update @delete
  {
    crule_id
    dbt_rules @unnest
  }
}
```

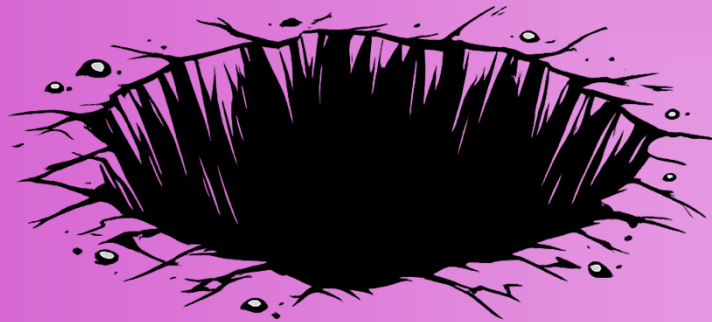
```
rule_id @nocheck
rule_name @nocheck
dbt_tenants @unnest
{
  ten_id @nocheck
  ten_name @nocheck
}
dbt_severities @unnest
{
  sev_id @nocheck
  sev_name @nocheck
}
dbt_severity_levels @unnest
{
  sev_id @nocheck
  sev_name @nocheck
}
custom_parameters: dbt_custom_parameters
  @insert @update @delete
{
  cparam_id
  dbt_parameters @unnest
  {
    param_id @nocheck
    param_name @nocheck
    dbt_tenants @unnest
    {
      ten_id @nocheck
      ten_name @nocheck
    }
  }
  cparam_value
}
ignore_test_results: dbt_ignore_test_results
  @insert @update @delete
{
  ign_id
  dbt_tests @unnest
  {
    test_id @nocheck
    test_name @nocheck
    dbt_dbms_scopes @unnest
    {
      rdb_id @nocheck
      dbt_dbms @unnest
      {
        dbms_id @nocheck
        dbms_name @nocheck
      }
    }
    dbt_rules @unnest
    {
      rule_id @nocheck
      rule_name @nocheck
      dbt_tenants @unnest
      {
        ten_id @nocheck
        ten_name @nocheck
      }
    }
  }
  ign_identifier
  ign_comment
}
};
```

Duality View Based on 48 Tables





#1 Size



Size Matters



JSON Datatype Size Limits

32 MB

33'554'432 Bytes

JSON

(Oracle's native binary JSON format)

JSON Size - Examples

```
with j (variant, json_string) as (values
  ('rows_as_objects', '
    {
      "emps": [ { "empno": 7788, "name": "Scott", "sal": 3000 },
                  { "empno": 7839, "name": "King", "sal": 5000 },
                  { "empno": 7876, "name": "Adams", "sal": 1100 } ]
    }
  '),
  ('rows_as_arrays', '
    {
      "emps": [ [ 7788, "Scott", 3000 ],
                  [ 7839, "King", 5000 ],
                  [ 7876, "Adams", 1100 ] ]
    }
  ')
)
select variant,
  lengthb(json_string) as str_len,
  lengthb(replace(json_string, ' ', null)) as str_trimmed_len,
  lengthb(json(json_string)) as oson_len,
  lengthb(json(replace(json_string, ' ', null))) as oson_trimmed_len
from j;
```

VARIANT	STR_LEN	STR_TRIMMED_LEN	OSON_LEN	OSON_TRIMMED_LEN
rows_as_objects	242	138	125	125
rows_as_arrays	170	75	96	96

OSON Storage Representation

Variant	Offset	Hex	Bytes	ASCII Chars

rows_as_objects	000000	FF 4A 5A 01 21 06 04 00 14 00 50 00 01	42 6F BA	.JZ.!.....P.Bo.
	000010	E6 00 00 00 10 00 05 00 0B 04 65 6D 70 73 05 65		emps.e
	000020	6D 70 6E 6F 04 6E 61 6D 65 03 73 61 6C 84 01 01		mpno.name.sal...
	000030	00 05 C0 03 00 0D 00 25 00 3A 86 03 03 04 02 00		%.:.....
	000040	18 00 1C 00 22 22 C2 4E 59 05 53 63 6F 74 74 21		".NY.Scott!
	000050	C2 1F 9C 00 0D 00 2E 00 32 00 37 22 C2 4F 28 04		2.7".O{.
	000060	4B 69 6E 67 21 C2 33 9C 00 0D 00 43 00 47 00 4D		King!.3....C.G.M
	000070	22 C2 4F 4D 05 41 64 61 6D 73 21 C2 0C		".OM.Adams!..
rows_as_arrays	000000	FF 4A 5A 01 21 06 01 00 05 00 4B 00 00	42 00 00	.JZ.!.....K..B..
	000010	04 65 6D 70 73 84 01 01 00 05 C0 03 00 0D 00 22		.emps....."
	000020	00 36 C0 03 00 15 00 19 00 1F 22 C2 4E 59 05 53		.6.....".NY.S
	000030	63 6F 74 74 21 C2 1F C0 03 00 2A 00 2E 00 33 22		cott!.....*...3"
	000040	C2 4F 28 04 4B 69 6E 67 21 C2 33 C0 03 00 3E 00		.O(.King!.3...>.
	000050	42 00 48 22 C2 4F 4D 05 41 64 61 6D 73 21 C2 0C		B.H".OM.Adams!..

OSON or JSON Datatype Limitation?



OSON BLOB

```
create table i (  
  data blob check (data is json format oson)  
);  
  
exec dbms_random.seed(42);  
  
insert into i (data)  
select json_arrayagg(  
  json{  
    '_id' : rownum,  
    'name' : initcap(dbms_random  
      .string('A', 30)),  
    'sal' : trunc(dbms_random  
      .value(1000, 5000), 2)  
  } returning blob  
)  
from dual connect by level <= 750000;
```

ORA-40604: exceeded maximum size for a JSON data type object
JZN-00092: JSON too large for binary encoding



Conventional BLOB

```
create table i (  
  data blob check (data is json)  
);  
  
exec dbms_random.seed(42);  
  
insert into i (data)  
select json_arrayagg(  
  json{  
    '_id' : rownum,  
    'name' : initcap(dbms_random  
      .string('A', 30)),  
    'sal' : trunc(dbms_random  
      .value(1000, 5000), 2)  
  } returning blob  
)  
from dual connect by level <= 750000;
```

```
select length(data) from i;
```

```
LENGTH(DATA)  
-----  
51548929
```

Solutions for ORA-40604/JZN-00092?



Process JSON Chunks < 32 MB

```
create json relational duality view dv as
t @insert @update @delete
{
  _id: empno,
  name: name,
  sal: sal
};

insert into dv (data)
select jt.emp
from i
cross apply json_table(
  i.data,
  '$[*]' columns (
    emp json path '$'
  )
) jt;
```



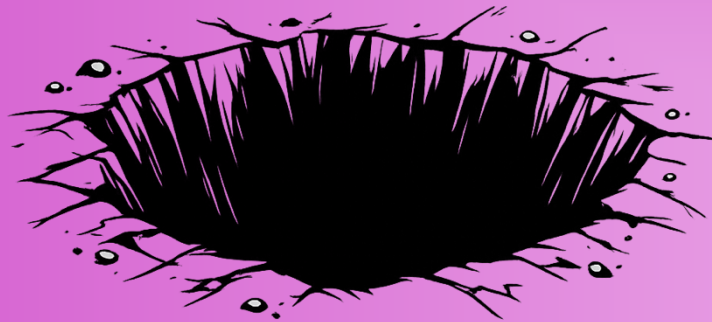
Avoid OSON and DVs

```
insert into t (empno, name, sal)
select jt.empno, jt.name, jt.sal
from i
cross apply json_table(
  i.data,
  '$[*]' columns (
    empno number      path '$._id',
    name varchar2(30 char) path '$.name',
    sal number(7,2)    path '$.sal'
  )
) jt;
```



Faster,
Max. 32 TB
with 8 KB Blocks

#2 Surrogate Keys



Surrogate Keys – Overview

Typical Characteristics

- System generated (Sequence, UUID)
- Not derived from other business values
- Never reused
- Not manipulable by the user or system
- No semantic meaning
- Exposed to the user or application for reference purposes only
- Not shared across systems

Values

- Single-column PKs and FKs (no composite and compound keys)
- Simplifies access and joins
- No PK and FK updates
- Simplifies BK updates
- Might improve join performance

Model

Tables

```
create table d (  
  dept_id integer generated always as identity not null,  
  dept_name varchar2(30 char) not null,  
  dept_desc varchar2(100 char),  
  constraint d_pk primary key (dept_id),  
  constraint d_uk unique (dept_name)  
);  
  
create table e (  
  emp_id integer generated always as identity not null,  
  emp_no integer not null,  
  emp_name varchar2(30 char) not null,  
  dept_id integer not null,  
  constraint e_pk primary key (emp_id),  
  constraint e_uk1 unique (emp_no),  
  constraint e_uk2 unique (emp_name),  
  constraint e_d_fk foreign key (dept_id) references d  
);
```

Duality View (PK)

```
d @insert @update @delete  
{  
  _id: dept_id,  
  dept_name,  
  dept_desc  
  emps: e @insert @update @delete  
  {  
    emp_id,  
    emp_no,  
    emp_name  
  }  
};
```

Ensures that
sequences remain
unique

Model enforces
server- side
surrogate keys

JSON for _id 2

```
{
  "_id" : 2,
  "_metadata" :
  {
    "etag" : "DFA31E21B395F4196E69AE43F8E692B9",
    "asof" : "00000000006AFCE2"
  },
  "dept_name" : "Research",
  "dept_desc" : "Chicago",
  "emps" :
  [
    {
      "emp_id" : 2,
      "emp_no" : 7566,
      "emp_name" : "Jones"
    },
    {
      "emp_id" : 3,
      "emp_no" : 7788,
      "emp_name" : "Scott"
    }
  ]
}
```

Good for
original system

Not optimal for
sharing across
systems

Duality View Based on Business Keys?

Duality View (BK)

```
create json relational duality view dv_bk as  
d @insert @update @delete  
{  
  _id: dept_name,  
  dept_desc,  
  emps: e @insert @update @delete  
  {  
    emp_no,  
    emp_name  
  }  
};
```

ORA-42550: Cannot create updatable JSON-relational duality view
'DV_BK': linking column 'DEPT_ID' of table 'E' should be selected.

- no FK columns based on BK
- Missing relationship between D and E
- Read-only DV is possible
- Updatable DV on BK would require redundant data
- Updatable DV on BKs are only feasible if BKs are used exclusively for PKs and FKs

Solution – 1/3) Update DV

Duality View (PK, BK)

```
create or replace json relational duality view dv as
d @insert @update @delete
{
  _id: dept_id
  dept_name
  dept_desc
  emps: e @insert @update @delete
  {
    emp_no
    emp_name
  }
};
```

- Remove PK emp_id
- Works when not used as FK
- Object identified via BK (UK)
- But we have two UKs
 - emp_no
 - emp_name

Solution – 2/3) Remove PK, Metadata

Query

```
select json_serialize(  
  json_transform(  
    data,  
    remove '$._id',  
    remove '$._metadata'  
  ) pretty  
  ) as export_json  
from dv dv where dv.data.dept_name = 'Research';
```

Result

```
{  
  "dept_name" : "Research",  
  "dept_desc" : "Chicago",  
  "emps" :  
  [  
    {  
      "emp_no" : 7566,  
      "emp_name" : "Jones"  
    },  
    {  
      "emp_no" : 7788,  
      "emp_name" : "Scott"  
    }  
  ]  
}
```

Solution – 3/3) Add PK on Import

Import Procedure

```
create or replace procedure import(in_json in json) is
  cursor c_dept is
    select dept_id
      from d
     where dept_name = json_value(in_json, '$.dept_name');
  r_dept c_dept%rowtype;
  cursor c_transform is
    select json_transform(
      in_json,
      set '$._id' = r_dept.dept_id create on missing
    ) as data;
  r_transform c_transform%rowtype;
begin
  open c_dept;
  fetch c_dept into r_dept;
  close c_dept;
  open c_transform;
  fetch c_transform into r_transform;
  close c_transform;
  if r_dept.dept_id is not null then
    update dv dv
      set data = r_transform.data
     where dv.data."_id" = r_dept.dept_id;
  else
    insert into dv (data) values (r_transform.data);
  end if;
end import;
```

Call

```
begin
  import(json('
    {
      "dept_name": "Research",
      "dept_desc": "Chicago2",
      "emps": [
        {
          "emp_no": 7566,
          "emp_name": "Jones2"
        },
        {
          "emp_no": 7788,
          "emp_name": "Scott2"
        }
      ]
    }
  '));
end;
```

```
select * from e natural join d;
```

```
DEPT_ID EMP_ID EMP_NO EMP_NA DEPT_NAME DEPT_DESC
```

```
-----
1      1  7839 King   Accounting New York
2      4  7566 Jones2 Research  Chicago2
2      5  7788 Scott2 Research  Chicago2
```

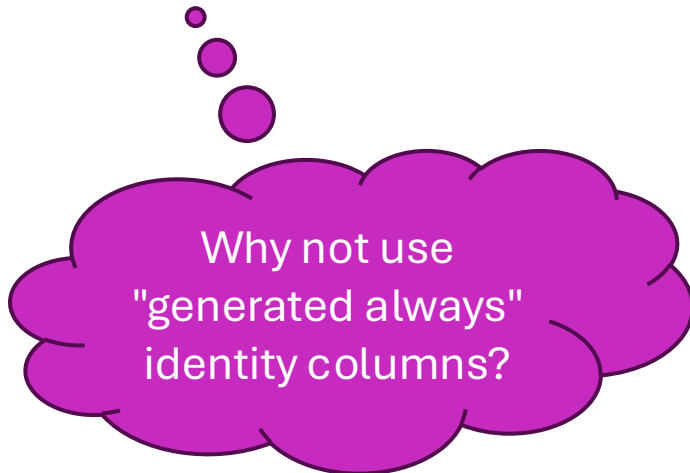
Delete/Insert for
each emp due to
UK update

First UK by
property name is
used

Solution Approach in dbLinter Repo

Model

- Updateable DVs based on PK/FK
- All PK based on surrogate keys with default on null for insert only sys_guid()



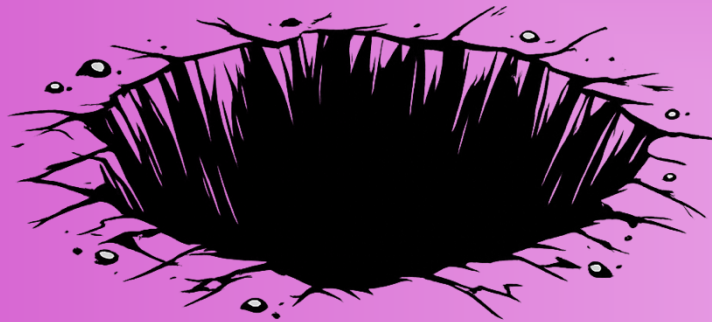
Export

- Remove _id node
- Remove _metadata node
- Remove all surrogate keys

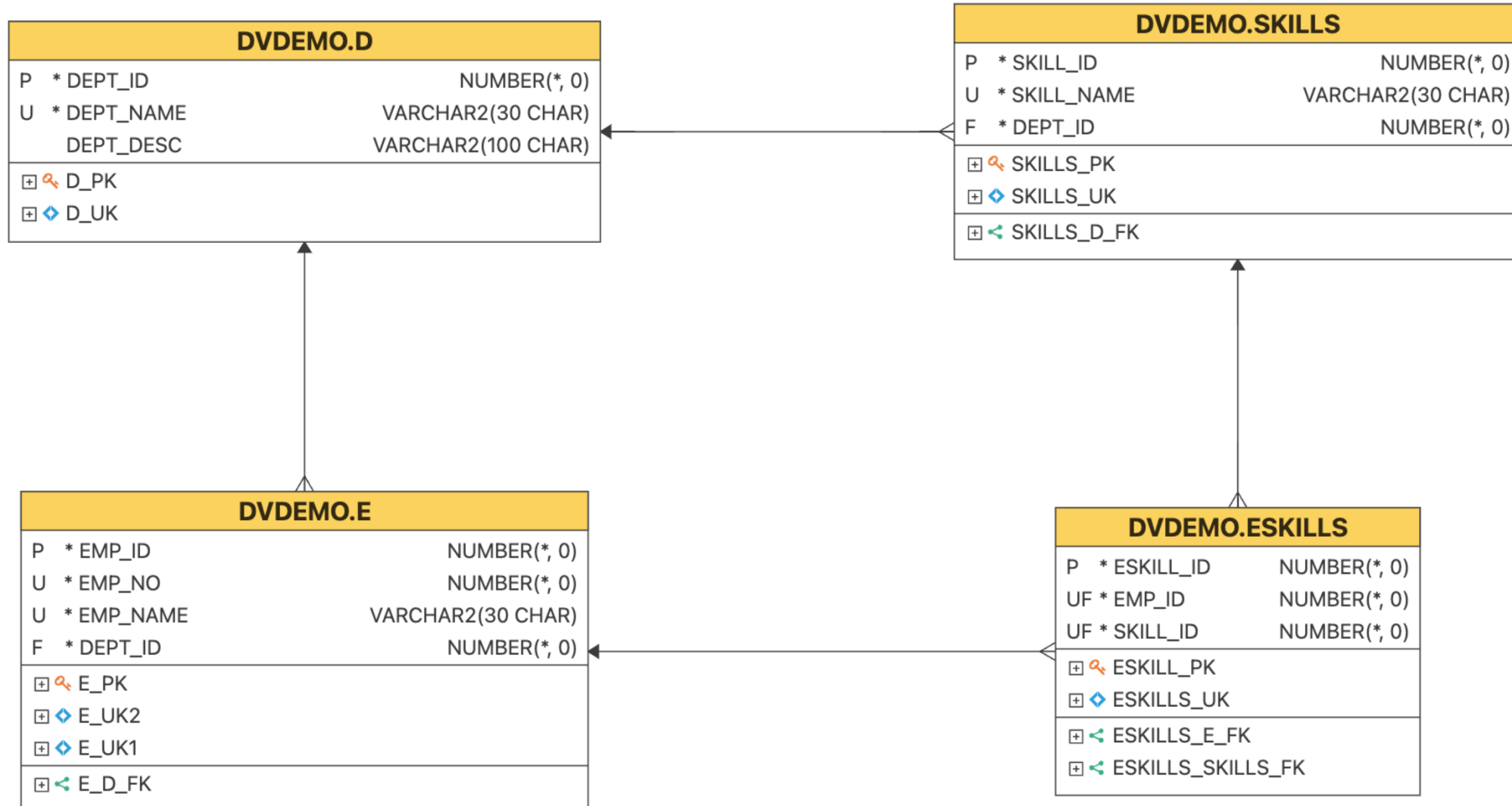
Import

- Populate surrogate keys via BK (an explicitly chosen UK)
- Insert via DV (no _id)
- Update via DV (existing _id)

#3 Referential Integrity



Model – Overview



Model – Sequence, Tables

```
create sequence ids;

create table d (
  dept_id integer default on null
           for insert only ids.nextval,
  dept_name varchar2(30 char) not null,
  dept_desc varchar2(100 char),
  constraint d_pk primary key (dept_id),
  constraint d_uk unique (dept_name)
);

create table skills (
  skill_id integer default on null
             for insert only ids.nextval,
  skill_name varchar2(30 char) not null,
  dept_id integer not null,
  constraint skills_pk primary key (skill_id),
  constraint skills_uk unique (skill_name),
  constraint skills_d_fk foreign key (dept_id)
                        references d
);
```

```
create table e (
  emp_id integer default on null
              for insert only ids.nextval,
  emp_no integer not null,
  emp_name varchar2(30 char) not null,
  dept_id integer not null,
  constraint e_pk primary key (emp_id),
  constraint e_uk1 unique (emp_no),
  constraint e_uk2 unique (emp_name),
  constraint e_d_fk foreign key (dept_id) references d
);

create table eskills (
  eskill_id integer default on null
                for insert only ids.nextval,
  emp_id integer not null,
  skill_id integer not null,
  constraint eskill_pk primary key (eskill_id),
  constraint eskills_uk unique (emp_id, skill_id),
  constraint eskills_e_fk foreign key (emp_id)
                        references e,
  constraint eskills_skills_fk foreign key (skill_id)
                        references skills
);
```

Duality View

```
create or replace json relational duality view dv as
d @insert @update @delete
{
  _id: dept_id
  dept_name
  dept_desc
  skills: skills @insert @update @delete
  {
    skill_id
    skill_name
  }
}
emps: e @insert @update @delete
{
  emp_id
  emp_no
  emp_name
  skills: eskills @insert @update @delete
  {
    eskill_id
    skills @unnest
    {
      skill_id
      skill_name
      d @unnest
      {
        dept_id @nocheck
        dept_name @nocheck
      }
    }
  }
}
};
```

Insert into DV

```
insert into dv (data) values (json('{
  "_id" : 1,
  "dept_name" : "HR",
  "dept_desc" : "Human Resources",
  "skills" :
  [
    {
      "skill_id" : 2,
      "skill_name" : "SQL"
    },
    {
      "skill_id" : 3,
      "skill_name" : "AI"
    }
  ],
  "emps" :
  [
    {
      "emp_id" : 4,
      "emp_no" : 1234,
      "emp_name" : "Luma",
      "skills" :
      [
        {
          "eskill_id" : 5,
          "skill_id" : 3,
          "skill_name" : "AI",
          "dept_id" : 1,
          "dept_name" : "HR"
        }
      ]
    }
  ]
}'));
```

ORA-42692: Cannot insert into JSON Relational Duality View 'DVDEMO'. 'DV': Error while inserting into table 'ESKILLS'
ORA-02291: integrity constraint (DVDEMO.ESKILLS_SKILLS_FK) violated - parent key not found

Solution – Check Integrity At Commit

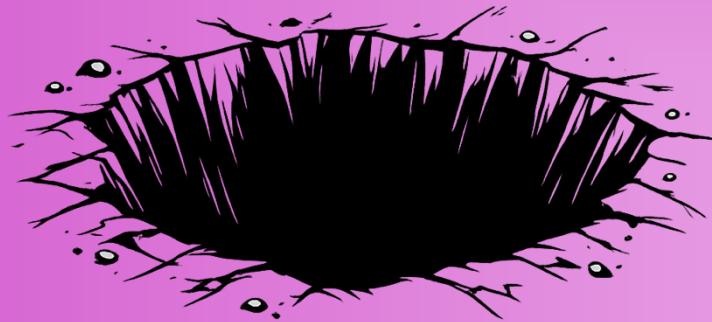
```
alter table eskills drop constraint eskills_skills_fk;  
alter table skills drop constraint skills_pk;  
  
alter table skills  
  add constraints skills_pk  
    primary key (skill_id) rely;  
  
alter table eskills  
  add constraint eskills_skills_fk  
    foreign key (skill_id) references skills  
    deferrable initially immediate rely;  
  
set constraints all deferred;  
insert into dv (data) values (...);  
commit;
```

"deferrable"
changes default
behaviour of RELY!

May lead to different
execution plans!

We defer the check
for DV DMLs only

#4 Composition vs. Aggregation



Composition

Dept with Emps

```
{
  "_id" : 2,
  "dept_name" : "Research",
  "dept_desc" : "Chicago",
  "emps" :
  [
    {
      "emp_id" : 4,
      "emp_no" : 7566,
      "emp_name" : "Jones"
    },
    {
      "emp_id" : 5,
      "emp_no" : 7788,
      "emp_name" : "Scott"
    }
  ]
}
```

- Single document structure
- Linked life cycle
- Limited update flexibility

Aggregation



Departments

```
{
  "_id" : 2,
  "dept_name" : "Research",
  "dept_desc" : "Chicago"
}
```



Employees

```
{
  "_id" : 4,
  "emp_no" : 7566,
  "emp_name" : "Jones",
  "dept_id" : 2,
  "dept_name" : "Research"
}

{
  "_id" : 5,
  "emp_no" : 7788,
  "emp_name" : "Scott",
  "dept_id" : 2,
  "dept_name" : "Research"
}
```

- Multiple document structure
- Independent life cycle
- Update flexibility

One Model – Multiple Representations

Composition

```
create json duality view dv as
d @insert @update @delete
{
  _id: dept_id,
  dept_name,
  dept_desc
  emps: e @insert @update @delete
{
  emp_id,
  emp_no,
  emp_name
}
};
```

Aggregation

```
create json duality view d_dv as
d @insert @update @delete
{
  _id: dept_id
  dept_name
  dept_desc
};

create json duality view e_dv as
e @insert @update @delete
{
  _id: emp_id
  emp_no
  emp_name
  d @unnest
{
  dept_id
  dept_name
}
};
```

Composition Dept with Emps

Operation	emps Array	emps @insert @update @delete	emps @noinsert @nouupdate @nodelete
Insert	emps: []	Deletes emps	Keeps emps (deferred or disabled constraints)
Update	emps: []	Deletes emps	Keeps emps
Update	emps: [{Jones}]	Deletes emps except Jones Insert or Updates Jones	ORA-40938: Cannot delete from table 'E' in JSON Relational Duality View 'DV': Missing DELETE annotation or NODELETE annotation specified.
Delete	-	Deletes emps	Succeeds for depts without emps ORA-40938: Cannot delete from table 'E' in JSON Relational Duality View 'DV': Missing DELETE annotation or NODELETE annotation specified.

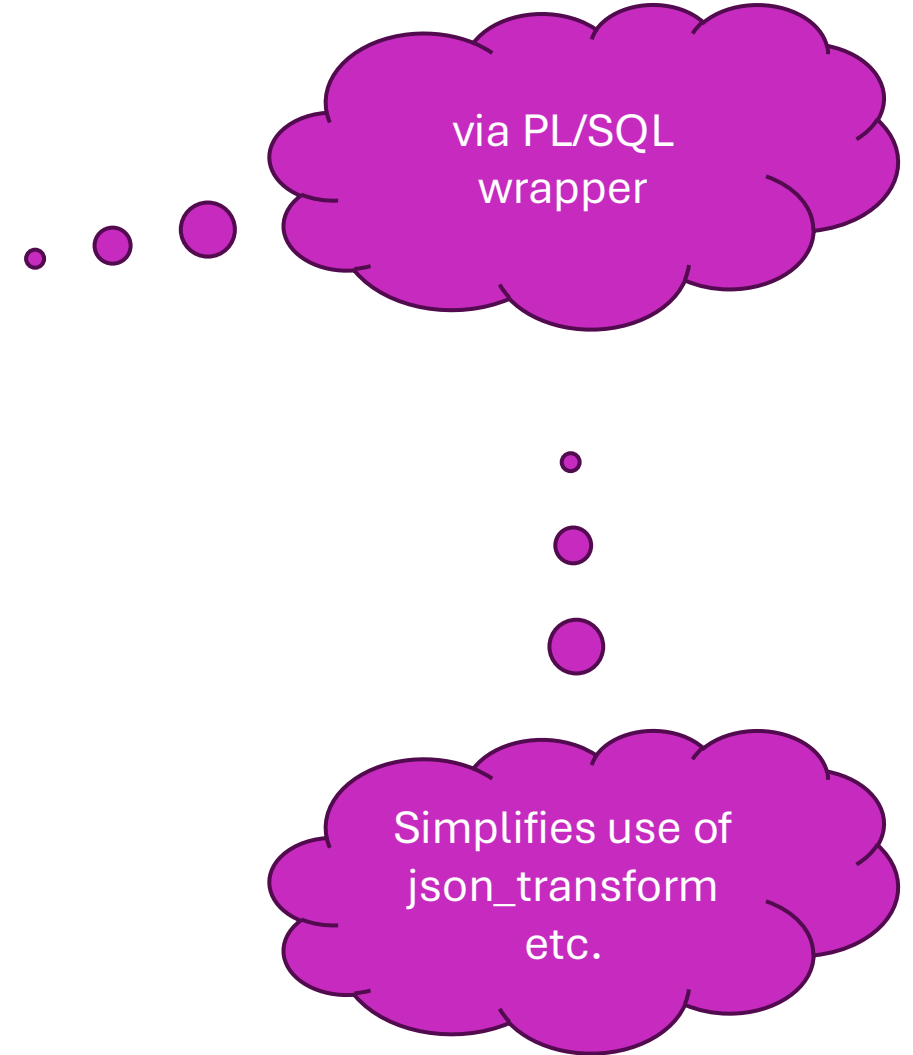
other annotation combinations?

... rarely relevant

... also due to technical delete/insert operations instead of a single update

JSON in dbLint Repo

- Duality Views for Export & Import
 - Tenants
 - Users
- Duality Views for Export only
 - Rules (for <https://dblint-rules.united-codes.com/>)
 - Logs
- SQL/JSON for read-only access
 - Open Client Session



Core Messages

JSON Documents in Oracle AI Database

- Keep JSON documents in duality views small
- Use SQL/JSON for large JSON documents
- Consider use case-specific (duality) views / JSON documents
- Defer constraint checks for embedded relationships in updatable duality views



JSON Documents Shared Across Systems

- Remove surrogate keys from source system
- Add surrogate keys from target system based on business keys
- Never use "generated always" identity columns for embedded relationships



Thank you!

